第8章

メモリ・カードのWAVファイルを再生する

ディジタル・オーディオ・プレーヤの製作

河野 崇

ここでは，付属FPGA基板を使ったディジタル・オーディオ・プレーヤの設計事例を取り上げる．SDメモリ・カードに書き込んだwav形式の音声データを再生することができる．データ処理には自作のCPUコアを利用している．　（編集部）

ファイル・システムを持った記憶媒体から音楽を再生するディジタル・オーディオ・プレーヤが一般向けに販売されるようになってから10年が経ちます．この間，ハード・ディスクやフラッシュ・メモリの大容量化，低価格化が進み，ディジタル音楽プレーヤが広く普及する一つの要因となりました．特にここ数年のフラッシュ・メモリの急激な低価格化には目を見張るものがあります．実用的な容量のシリコン・ディスクが，まだ高めとはいえ現実的な価格で入手可能となるほどです．フラッシュ・メモリを用いたメモリ・カードも手軽に利用できるようになりました．

携帯型ディジタル・オーディオ・プレーヤでは，小型の装置の中に大量の楽曲を収録するため，非可逆圧縮をかけて利用しています．しかしピュア・オーディオの世界にも音楽CDメディアの代替としてメモリ・カードを持ち込み，CDあるいはそれ以上の品質の無圧縮楽曲を再生するプレーヤがあってもよい時期になってきたと思います．パソコンとの情報のやりとりもより簡単になり，より手軽に，より大容量の「マイCD」を作ることができるようになります．

そこで今回は，入手性に優れたSDメモリ・カードを用い，ピュア・オーディオ・クラスの音質を視野に入れたディジタル・オーディオ・プレーヤを製作しました．

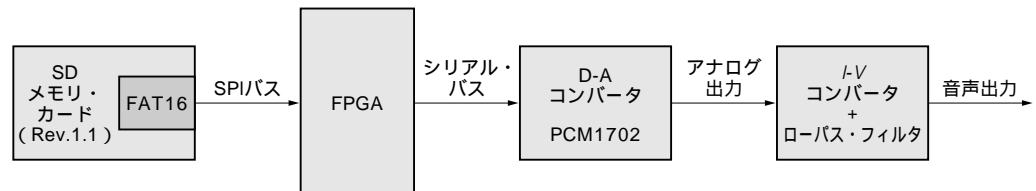
1．仕様を決めFPGA周辺回路を設計する

製作したオーディオ・プレーヤのシステム構成を**図1**に示します．

● 2GバイトまでのSDメモリ・カードに対応

今回は，コストや技術的容易さから2GバイトまでのSDメモリ・カード（Rev.1.1）を扱います．

SDメモリ・カードへのアクセス方法には，SDバス・モードとSPIバス・モードがあります．SDバス・モードは，4ビットのデータ・バスを利用することにより，最大25Mバイト/sの転送速度を実現しています．SPIバス・モードは，入出力それぞれ専用の1ビットのデータ線を用いることにより，最大6.25Mバイト/sが実現可能で，MMC（MultiMedia Card）との互換性があります．今回は

図1 システム構成
2GバイトまでのSDメモリ・カードを扱う．WAV形式で記録されているデータを再生する．20ビットD-Aコンバータにより音声出力を行う．

KeyWord　FPGA，SDメモリ・カード，SPIバス・モード，FAT16，PCM1702，OPA627AP，SC2004，CPUコア

仕様の公開状況から，SDメモリ・カードのアクセスにはSPIバス・モードを用います．

2GバイトまでのSDメモリ・カードのファイル・システムはFAT16です．4Gバイト以上のSDHC（High Capacity）カードではFAT32が採用されています．

サポートする音楽ファイルの形式は，音楽CDと同等の44.1kHz，16ビット・ステレオ，リニアPCMのwavファイルとします．

● 音声出力部には20ビットD-Aコンバータを利用

D-Aコンバータは，米国Texas Instruments社のBurr-Brown製品「PCM1702」を用いました．これは，ハイエンドの機器でも使われているPCM1704（24ビット）の弟分で，サイン・マグニチュード方式の20ビットD-Aコンバータです．エントリ・レベルの機器に使われており，入手性も悪くないICです．

ディジタル入力はシリアル・バスです．同じハードウェアで「PCM1704」や「PCM56（16ビット）」を接続できます．PCM1704やPCM1702は，CDのサンプリング周波数の16倍程度までの周波数での動作が保証されています．FPGA内部で16倍オーバサンプリングを行えば，最終段のローパス・フィルタをコンデンサと抵抗だけで構成することも可能です．

最終段が音質に決定的な影響を持っていることを考えると，音質に多大な寄与を行うことが明らかですが，今回はなるべく単純にするため，44.1kHzのままで出力しています．

● FPGA周辺回路を設計する

設計したプレーヤの全体回路図を**図2**に示します．製作した基板を**写真1**に示します．

SDメモリ・カードは，弱いプルアップ抵抗のみでFPGA

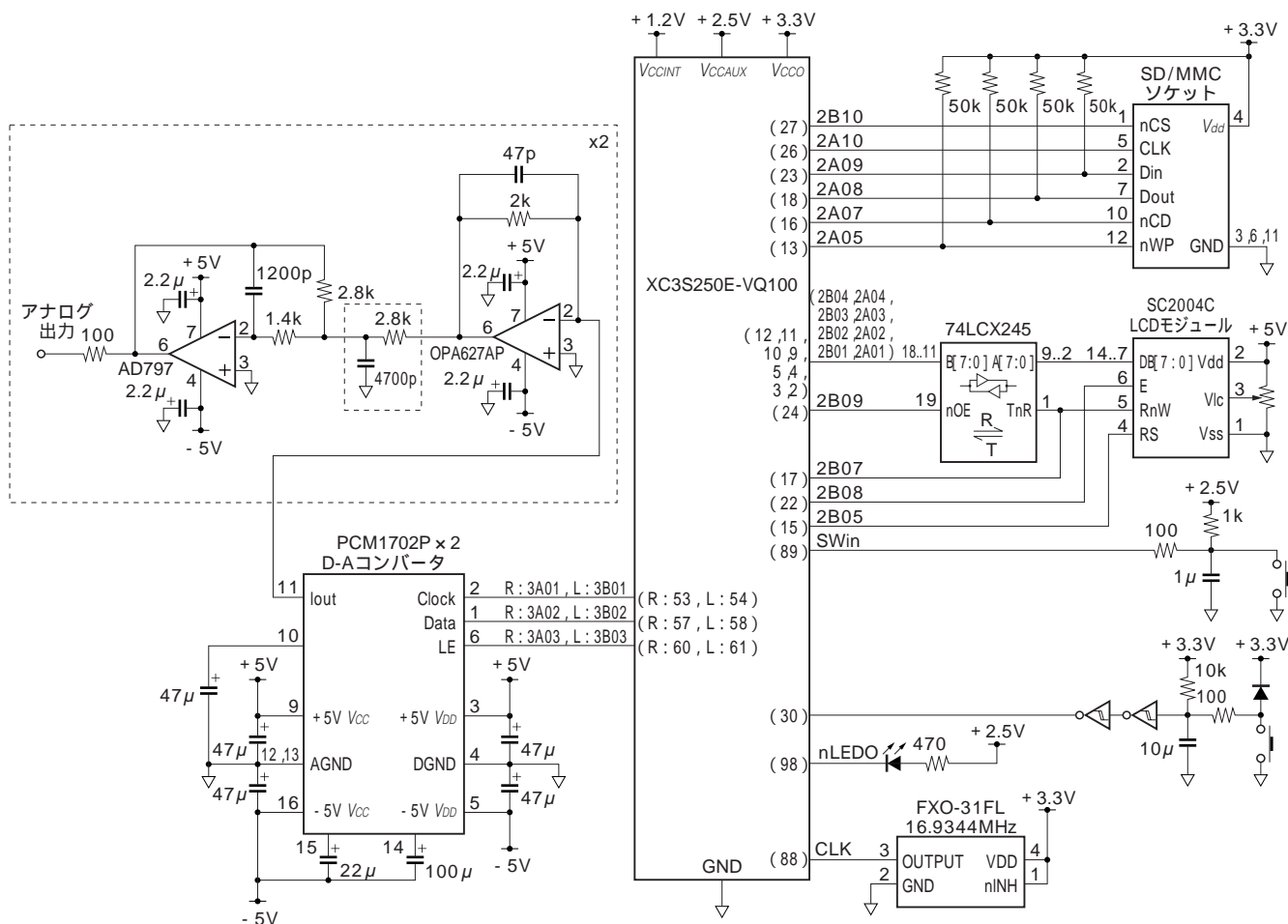



図2　製作したディジタル・オーディオ・プレーヤの回路図

の3.3V I/Oに直結できます．nCDとnWPは，カード・ソケット（サンハヤトの「CK-29」）に設置されたスイッチで，それぞれSDメモリ・カードが挿入されているか，ライト・プロテクト・スイッチがONかを知らせる信号です．回路図上は両方とも配線していますが，今回はリードしか行いませんのでnCDのみしか使いません．

D-AコンバータPCM1702の制御線は，Clock，Data，LEの3本です．左チャネルと右チャネル用に1組ずつ用意します．PCM1702は電流出力（±1.2mA）なので，*I-V*変換を行う必要があります．1本の抵抗器で変換できるのが理想ですが，PCM1702の内部に保護ダイオードが入っており，±0.6V以上の電圧を発生させることができないので，能動部品を使った*I-V*変換回路を付ける必要があります．今回は，Texas Instruments社のBurr-Brown製品「OPA627AP」を用いました[1]．

この後，アクティブ・ローパス・フィルタにより高周波ノイズをカットします．ローパス・フィルタに使用するOPアンプが特に音質に影響を与えやすいと言われているので，さまざまなものを試して自分の好みのものを探すのがよいと思います．また，電源や配線も大きく音質に影響を与える因子です．$\pm V_{DD}$，$\pm V_{CC}$は別電源とし，配線もなるべく左側（1～6ピン）と右側（9～16ピン）が分かれるように気を付けるとよいでしょう（今回の製作では共通電源としている）．配線については，プリント基板のサンプル設計[1]が参考になります．

クロックは44.1kHzの整数倍となる16.9344MHzを選びました．16倍オーバサンプリング，24ビット/ワードでD-Aコンバータにデータ送出するときに用いる周波数と同じです．

以上に加えて，20文字×4行のキャラクタ表示LCDモジュール「SC2004C」を接続しています．ファイル名表示やデバッグのために用いることを意図しています．このモジュールはI/Oが5Vなので，入力も行うデータ・バス（DB[7:0]）には電圧レベル変換が必要です．今回は標準ロジックICの「74LCX245」を用いました．

2．データ/音声処理回路をFPGAに実装する

FPGA内に実装する回路のブロック図を**図3**に示します．

● 設計資産を活用する

「luna0＋IO」は，自作のCPUコアにメモリとI/Oをアクセスする回路や割り込みコントローラを付けたものです．このコアは，8ビットのRISCライクなCPUであり，低コストFPGAで90MHz～110MHz前後の周波数で動作します[2][3]．

今回は16.9344MHzのクロックをPLL（DCM_SP）により5倍してシステム・クロックとしています．pct（タイマ・

写真1
製作したディジタル・オーディオ・プレーヤの外観

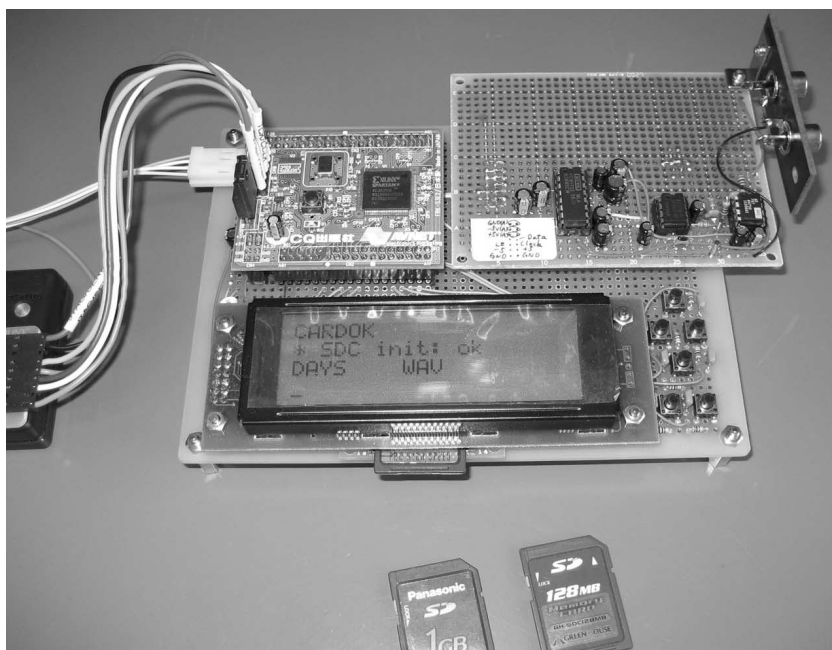
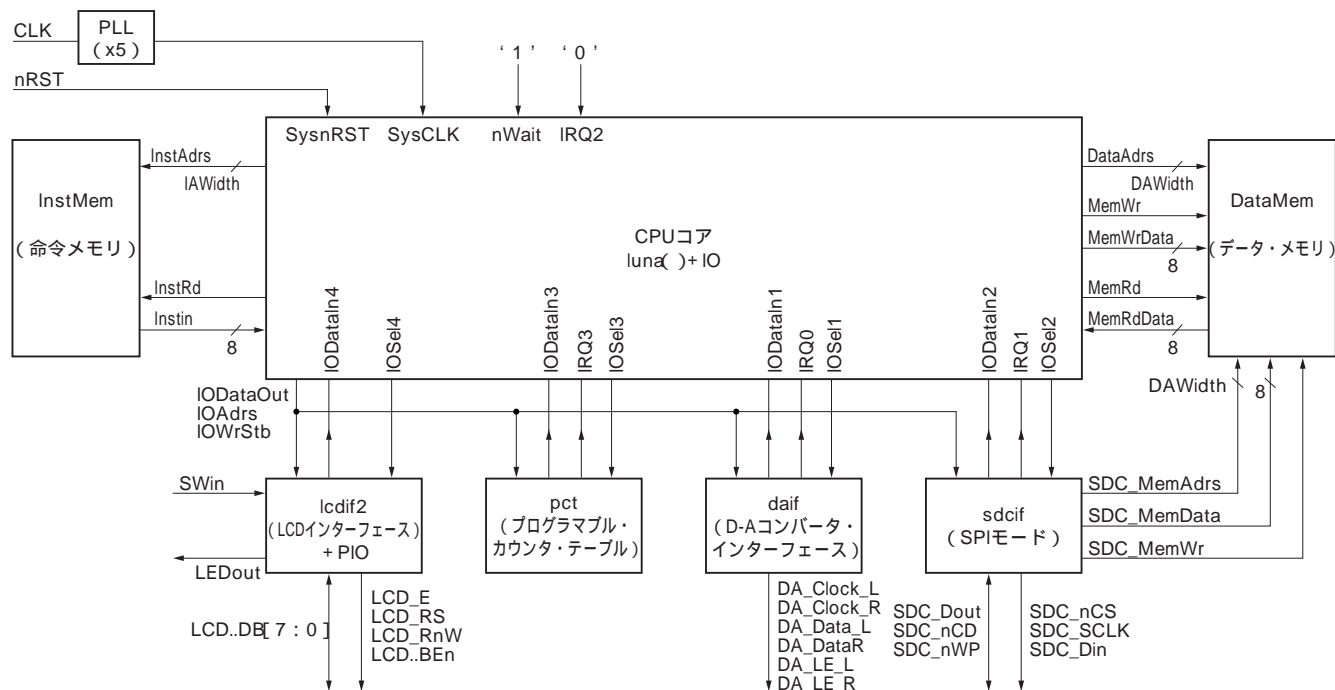



図3　FPGAに実装する回路のブロック図
自作のCPUコアを用いている．過去の設計資産を活用したので，今回新しく設計したのはsdcifとdaifだけである．

モジュール）も以前設計したもの(2)(3)と同じです．液晶モジュールへは，lcdif2ブロックを経由してアクセスします．これは以前設計したもの(2)(3)に，レベル変換バッファ（74LCX245）の制御線とパラレルI/Oを追加したものです．

今回新たに設計したのは，SDメモリ・カードをSPIモードで制御するためのモジュールsdcifと，D-Aコンバータへデータを送出するモジュールdaifの二つだけです．

各モジュールのI/Oアドレスは，800xhがl0pic（割り込みコントローラ），801xhがdaif，802xhがsdcif，803xhがpct，804xhがlcdif2です．

● SDメモリ・カードのプロトコルを理解する

SDメモリ・カードのSPIモードのアクセス方法を簡単に**図4**に示します(4)～(7)．

SPIモードはMMCと互換性があるので，MMCの技術文書も役に立ちます（SD Card Associationの文書は公開されていない部分が多いので，MMCの技術文書を参考にしないと設計できない）．

(1) 4本の信号でアクセスする

SPIモードでは，CLK，nCS，Din，Doutの4本の信号線を用います（**図4(a)**）．CLKはクロック，Dinは外部からSDメモリ・カードへの入力データ，DoutはSDメモリ・カードから外部への出力データです．いずれもアイドル状態では“H”です．nCSはアクティブ“L”のチップ・セレクト信号で，この信号がアサートされたクロックから8クロックごとに8ビットずつデータが解釈されます．通常の（High-Speed mode非対応の）カードではCLKは25MHz以下です．また，電源投入時には適切な初期化が終わるまで400kHz以下で駆動しなければなりません．DinはCLKの立ち上がり時にSDメモリ・カードが取り込みますが，セットアップ時間もホールド時間も同じくらいなので，ホストはCLKの立ち下がり時にデータをセットしてやります．DoutはCLKの立ち上がり時に取り込みます．

(2) SDメモリ・カードへのコマンド送信

トランザクション（オペレーションと呼ばれる）は，ホストがnCSをアサートし，それから8の倍数回目（0も可）のクロックでDinを‘0’に駆動することで始まります．**図4(b)**に示すように，ホストはDinの最初のビットを‘0’にした後いったん‘1'に戻し，それから6ビットのコマンド・インデックス（コマンド内容を指示する），32ビットの引き数，7ビットのCRCコードを送出し，最後に‘1'に駆動します（計48ビット）．いずれもMSBから先に送信します．SDメモリ・カードのコマンドはCMD0やCMD17など，このコマンド・インデックスの値で呼ばれます．

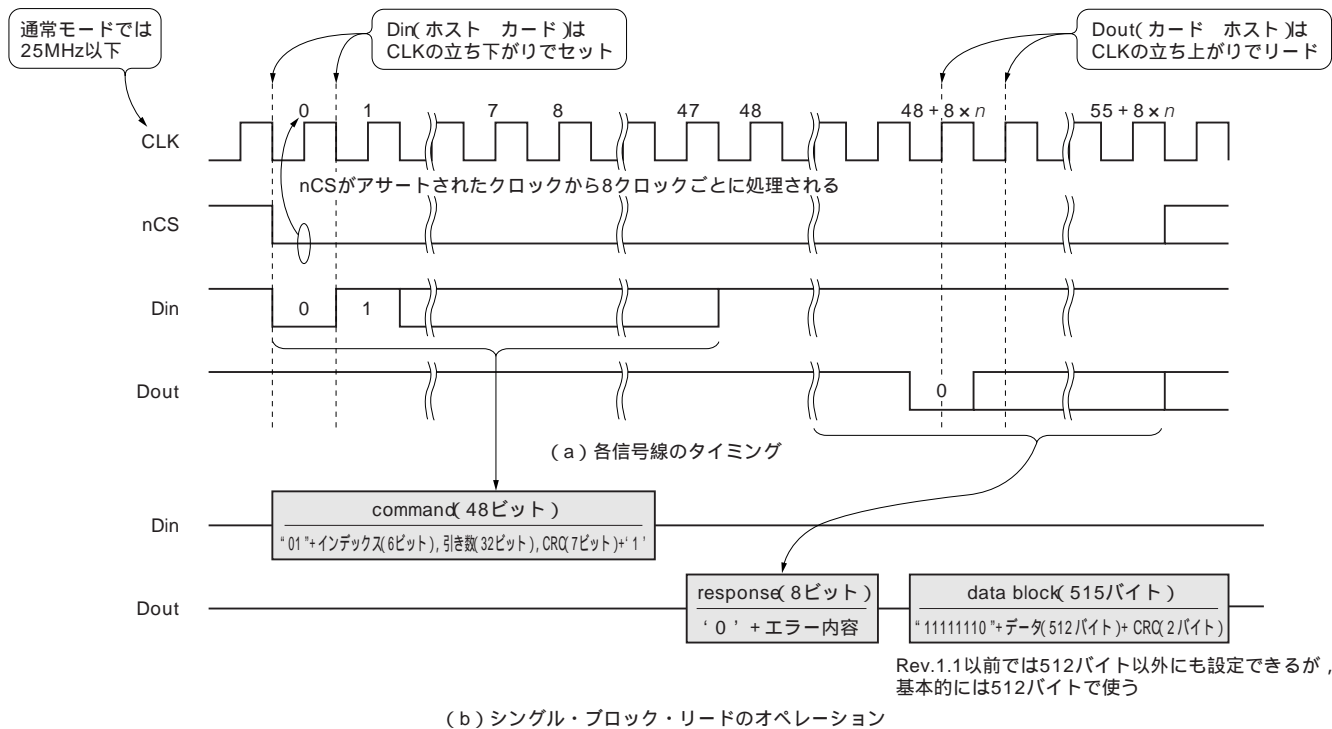



図4　SDメモリ・カードのSPIモードのプロトコル
SPIモードはMMC(MultiMedia Card)と互換性がある.

(3) SDメモリ・カードのコマンドへの応答

コマンドを受けたSDメモリ・カードは,nCSがアサートされてから8の倍数回目のクロックでDoutを'0'にし,続いて7ビットのデータを送信します(response).この7ビットはエラー情報で,すべて'0'ならば正常にコマンドが受け付けられたことを意味します.

コマンドによって7ビットではなく,15ビットや39ビットとより長いresponseの場合もあります.いずれもMSBより送信されてきます.データ転送を伴わないコマンドの場合はこれで終了です.

(4) SDメモリ・カードからのデータの読み出し

リード(CMD17)の場合は,SDメモリ・カードは要求されたデータを用意します.そしてnCSがアサートされてから8の倍数回目のクロックでDoutに"11111110"というバイト(Start Block),下位アドレスから512バイトのデータ(各バイトはMSBが先),16ビットのCRCの順で続けて送信します(**図4(b)**).

これら一連のデータをデータ・ブロックと呼びます.複数のデータ・ブロックを連続して送信させるコマンドも存在します(CMD18).

(5) SDメモリ・カードへのデータの書き込み

ライト(CMD24)の場合は,responseを受けたホストが,nCSのアサートから8の倍数回目のクロックでDinに"11111100"というバイト(Start Block Token),512バイトのデータ,16ビットのCRCの順で送信します.しばらくするとnCSのアサートから8の倍数回目のクロックで,SDメモリ・カード側からデータの受け入れ状況を報告するdata_responseと呼ばれる8ビットのデータが送られてきます.この後,書き込み処理の間"00000000"(Busy Token)が送り続けられ,新たなコマンドを受け付けられないことを示します.

リードと同様,複数のデータ・ブロックを連続して送信するコマンドも存在します(CMD25).

Ver. 1.1ではデータ・ブロックに含まれるデータの数を変更するコマンドも存在しますが,Ver. 2.0以降ではサポートされなくなるため,512バイトに固定しておいた方がよいでしょう.

● SDメモリ・カード・インターフェースsdcifの設計

今回はデータの書き込みは必要ありません.sdcifは,データなしのコマンドかリード・オペレーションを行いま

す．**図5**に状態遷移図を示します．

ハードウェア・リセットがかかっている期間以外は常にCLKと，CLKの立ち下がり8回ごとに1システム・クロック間アクティブになるPDLdという信号を生成しています．データの送受信（シリアル-パラレル変換），状態遷移はPDLdに同期して行います．

電源投入時はIdleステートにいます．スタート・コマンドを受けるとCmndステートに移行，6バイトのコマンドを送出後，Respステートでレスポンスを待ちます．responseを受信した場合，データなしの場合はTRNDステートへ移行します．データ・リードの場合はWDRdステートに移行して“11111110”（Start Block）を待ち，受信したらDtRdステートに移行して512バイト分のデータを受け取ります．このデータは直接データ・メモリ（デュアルポート）に転送します．この後はCRC1，CRC2ステートでCRCを読み捨てて，TRNDステートへ移行します．複数ブロック転送の場合は，CRC2ステートからWDRdに戻りますが，今回は性能上必要なかったので実装していません．

TRNDステートは，ダミー・クロックの期間を長くとるために用意しました．本来はIdleステートの最低8クロックだけで足りるので，なくてもかまいません．

表1に，CPUから見たsdcifのレジスタ構成を示します．CPUはまず，Configレジスタを設定し，オペレーション終了時に割り込みを発生させるかどうか（ビット7），SPIバスのCLKの速度（低速時は250kHz程度，高速時は14MHz程度），responseのタイムアウトを許可するかを指定します．次に，Command FIFOに送出する48ビットのコマンドを上位バイトから順に1バイトずつ書き込みます．

図5　sdcifのステート・マシン
状態遷移は，SPIのクロックの8回の立ち下がりに1回ずつ（システム・クロックの間），アクティブになる信号PDLdに同期して発生する．

表1 sdcifのレジスタ構成

アドレス（ベース・アドレスからの相対アドレス）	名 称	内 容	
+0	Control（W）	ビット7	予約
		ビット6	ソフトウェア・リセット（シーケンサとFIFOをリセット）
		ビット5	予約
		ビット4	リード・データあり
		ビット3	R1b
		ビット2..0	Response長－1（8ビット単位）
	Status（R）	ビット7	カードの有り無し（not nCD）
		ビット6	ソフトウェア・リセット中（‘1’でリセット動作中）
		ビット5	ライト・プロテクト（not nWP）
		ビット4..2	予約
		ビット1	resp（レスポンス受信済み）
		ビット0	Busy（動作中）
+1	Config（R/W）	ビット7	割り込み発生許可（‘1’=許可）
		ビット6	CLKモード（‘1’=高速，‘0’=低速）
		ビット5	タイムアウト許可（‘1’=許可）
+2	Command FIFO（W）	SDメモリ・カードに送出するコマンドを書き込む． 1回のオペレーションに必要な6バイトすべて書き込む． FIFOの内容がそのまま出力される．	
	Response FIFO（R）	responseで返された内容が入る．	
+4	DMA Address Low（W）	リード・データを転送するメモリ・アドレスの下位バイト	
+5	DMA Address High（W）	リード・データを転送するメモリ・アドレスの上位バイト	

データ・リード・コマンドの場合はDMA Addressレジスタに転送先アドレスをセットします．最後に，Controlレジスタに書き込み，オペレーション開始を指示します．ビット2～0には期待されるresponseの長さ－1，responseがR1bのときはビット3に'1'，リード・データを期待する場合はビット4に'1'を，ビット6に'0'を指定します．割り込みを使用しない場合はStatusレジスタのポーリングによりビット0のBusyをチェックしてオペレーション終了を待ちます．ビット6に'1'を指定した場合はオペレーション開始ではなく，ソフトウェア・リセット（FIFOのクリアと状態をIdleに戻す）となります．この場合はStatusレジスタのビット6をチェックして，ソフトウェア・リセットの終了を待つ必要があります．

● D-Aコンバータ用インターフェースdaifの設計

今回使用するD-Aコンバータは，Clock，Data，LEの3本からなるシリアル・バスによりデータを受け取ります．プロトコルを**図6**に示します．

Clockは20MHz以下のクロックです．DataはClockの立ち上がりエッジで取り込まれるシリアル・データで，上位ビットから順に送出します．LEはラッチ・イネーブル信号で，LSB送出時に'1'，その次のクロックで'0'を与えます．それ以外では'0'でも'1'でもかまいません．

PCM1702は20ビットのD-Aコンバータなので，LSBとその直前19ビット分のデータが取り込まれます．要は，Clockの周期ではなく，LEの立ち下がり周期がサンプリング周波数になるということです．daifでは24ビットごとにLEの立ち下がりを生成し，最初の4ビットをダミー・データ，次の16ビットをCPUから与えられたデータ，最後の4ビットを"0000"として出力しています．

表2にCPUから見たdaifのレジスタ構成を示します．左右チャネルそれぞれのデータ用にFIFOが用意されていま

図6 D-Aコンバータのインターフェース・プロトコル
Clock，Data，LEの3本からなるシリアル・バスである．

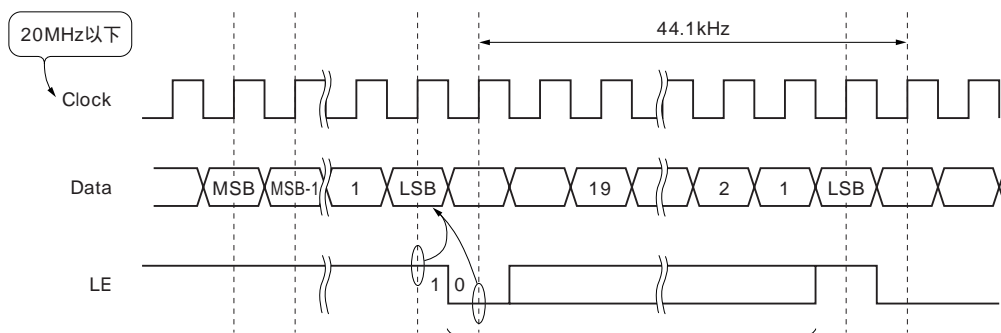



表2 daifのレジスタ構成

アドレス（ベース・アドレスからの相対アドレス）	名 称	内 容	
+0	Control（W）	ビット7..2	予約
		ビット1	Clear FIFOs
		ビット0	Enable（'1'）/Disable（'0'）
	Status（R）	ビット7	Almost Empty L
		ビット6	Almost Empty R
		ビット5	Empty L
		ビット4	Empty R
		ビット3	Full L
		ビット2	Full R
		ビット1	予約
		ビット0	Busy（動作中）
+1	Config（R/W）	ビット7	割り込み発生許可（'1'＝許可）
		ビット6..4	予約
		ビット3..0	Threshold－1（どちらかのFIFOの残りデータがThreshold Byte以下になると割り込みが発生）
+2	Data FIFO L（W）	Lチャネル用データFIFO．深さは16バイト＝8ワード	
+3	Data FIFO R（W）	Rチャネル用データFIFO．深さは16バイト＝8ワード	

す（Data FIFO L，　Data FIFO R）．各8ワード（16バイト）で，上位バイト，下位バイトの順に書き込みます．CPUはまず，Configレジスタで割り込みの設定を行います．

割り込みを使用する場合はビット7に‘1’を，ビット3～0に割り込みを発生させるThresholdから1を引いた値を設定します．どちらかのデータ用のFIFO内の残りデータ数がこの値以下になると割り込みが発生します．この割り込みは，残りデータがThreshold以下の間中，発生し続けるのではなく，Threshold＋1以上からThresholdに変化した時に発生します．従って，いったん残りデータ数をThre-shold＋1以上に持っていかないと割り込みは発生しませんし，発生後に割り込み要因のクリアなどの処理は必要ありません．また，FIFOの状態はStatusレジスタを読み出すことでもチェックできます．

ビット7とビット6はそれぞれLチャネルとRチャネルのデータ用FIFOの残りデータ数がThreshold以下の間，常に‘1’となります．ビット5とビット4はそれぞれLチャネルとRチャネルのデータ用FIFOが空の時，ビット3とビット2はいっぱいの時に‘1’になります．Configレジスタの設定が終わったらFIFOにデータを詰め，最後にControlレジスタのビット0に‘1’を書き込み，動作を開始させます．ビット1に‘1’を書き込むと両方のデータ用FIFOがクリアされます．sdcifとは異なり，動作中のみClockが生成されます．

● CPUコアで動作するソフトウェアの設計

CPUコアの処理として必要となる機能は，SDメモリ・カードからwavファイルを読み出し，daifに送出することです．

まず，SDメモリ・カードからデータを読み出す速度を検討します．再生に必要なデータ量は，

44.1［kHz］× 2［バイト］× 2［チャネル］
＝176.4［Kバイト/s］

です．SDメモリ・カードはデータ・ブロックの大きさは512バイトですから，

512［バイト］/176.4［Kバイト/s］＝ 2.902［ms］

分のデータを一度に読むことができることになります．今回テストに使ったメモリ・カード（容量128Mバイト）は，ロジック・アナライザで計測したところ，1データ・ブロックの読み出しに約0.8ms程度かかっていました．従って，1ブロックのデータが消費される間に少なくとも3ブロック分のデータを読み出すことができます．FATファイル・システムからwavファイルを読み出してとぎれなく再生するためには最低2ブロック読むことができなければなりません．これは，再生用データだけでなく，メモリ・カードのどの部分に必要なデータが入っているか記録したテーブル（FAT）をディスクから読み出さなければならないからです．もし，2ブロック読むことができない場合には，メモリ・カードのどの部分を読み出すかをあらかじめ計算してメモリに蓄えておく必要があります．今回使用したカードは古いものですが，この要件は十分に満たしていることが分かります．新しいメモリ・カードではアクセス速度は上がっていると思われますから，ファイルを再生しながら読み出し処理を行うことができることになります．

次に，daifに供給するデータについて考えてみます．D-Aコンバータに送るデータ速度は片チャネル当たり，

44.1［kHz］× 2［バイト］＝88.2［Kバイト/s］

です．FIFOは16バイトなので，16 / 88.2［Kバイト/s］で最低でも181.4μsに1回ずつdaifにアクセスする必要があるということが分かります．CPUコアのシステム・クロックは84.672MHzなので，15360クロックに1回の割り込みが発生します．CPUコアは，1クロックに1命令ずつ処理するので処理能力的には十分です．

以上から，**図7**のようにソフトウェアの構成を決めました．まず，メモリ・カードから読み込んだデータを1ブロック（512バイト）分保存するバッファを三つ用意します．そのうち一つはFAT用（SD_FAT_BUFFER），残りを再生データ用（DISK_BUFFER0，　DISK_BUFFER1）に割り当てます．daifのFIFOへのデータ充填は，daifの残りデータ数による割り込み発生機能（AlmostEmpty）を用い，割り込みプログラムによって行います．

このプログラムは，daifのFIFOの残りデータ数が2バイトになったときに呼び出され，6ワード分のデータを左右それぞれのチャネルに充填します．データはDISK_BUFFER0→DISK_BUFFER1→DISK_BUFFER0→…の順に参照します．この際，自分が次に読み出すデータが再生データ用バッファのどこにあるか示すアドレスを変数CPS_DA_DATAに，総再生データの残数（バイト単位）を変数RES_DA_DATAに保持，更新します．また，総再生

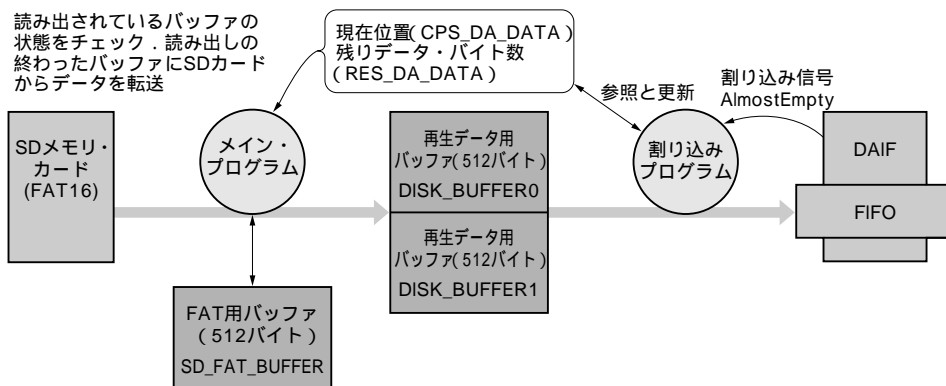



図7
再生ソフトウェアの構成
daifへのデータ送出は割り込みプログラムで，カードからの読み出しはメイン・プログラムで行う．

データ残数が0になったら，DISK_BUFFER{0,1}からのデータではなく，0をFIFOに充填するとともに，変数EMP_DA_DATAに1を設定して再生が終了したことを記録します．

電源が投入されたら，メイン・プログラムはFATファイル・システムのルート・ディレクトリから最初のファイルを探し，二つの再生データ用バッファ両方にデータを展開，再生開始位置を決定します．

● wavデータの再生

今回はリニアPCMのwavファイルのみをサポートします．この形式のファイルは，先頭から40バイト目から4バイトにバイト単位でのデータ数が格納されており，44バイト目が音楽データの先頭となっています．この領域のデータは時系列に従って，左16ビット，右16ビットの繰り返しとなっています．CPS_DA_DATAにこの開始位置アドレスを，RES_DA_DATAにデータ数を設定します．次にdaifのデータ用FIFOに最初の8ワード分のデータを充填し，割り込み許可，threshold＝2に設定します．さらに，割り込みコントローラに割り込みプログラムのアドレスを設定して割り込みマスクを解除してからdaifの動作を開始します．

後は常に変数CPS_DA_DATAを参照し，割り込みルーチンがDISK_BUFFER0とDISK_BUFFER1のどちらを参照しているかを監視し，参照バッファが切り替わったら参照終了したバッファに新しいデータを展開します．また，EMP_DA_DATAが1になったらdaifを停止します．

2曲目以降も同じ手順で再生できますが，時間的な都合で，本誌に収録されるデータでは1曲目だけの再生で終わっています．2曲目以降も再生できるように，今後アップデートする予定です．

● データ読み出しとFATファイル・システムの方法

SDメモリ・カードからのデータ読み出しと，FATファイル・システムの読み出しとを行うサブプログラムについて個別に解説を加えておきます．

(1) SDメモリ・カードからのデータ読み出し

SDメモリ・カードからのデータ読み出しは，sdc.asmおよびsdc_init.asmという二つのファイルに収められています(luna0アセンブラの文法については，参考文献[2][3]を参照)．

sdc_init.asmは，sdcifの初期化とともに，SDメモリ・カードの存在チェック(nCDのチェック)と初期化を行います．今回は必ずVer. 1.1(要は2Gバイト以下の)のSDメモリ・カードが挿入されるという仮定の下に，以下のように簡略化したシーケンスで行っています[4]．つまり，CMD0(GO_IDLE_STATE)でリセットをかけ，ACMD41(SD_SEND_OP_COND)をresponseが'00000000'で返ってくるまで繰り返します．ACMD41はCMD55に続けてCMD41を発行することをいいます．

ここで注意すべき点はCRCについてです．SPIモードのデフォルトでは，コマンドの最後のCRCは無視されるので，普段はCRCは必要ありません．しかし，電源投入直後はSDモードであるため，CMD0だけはCRCをつける必要があるようです[4]．CMD0は引き数が常に0なのでCRCは決めうちすることができ，エンドビットとあわせて95hを送ります．つまり，初期化プログラムが最初にsdcifのFIFOに送るデータは，40h, 00h, 00h, 00h, 00h, 95hとなります．

sdc.asmには，CMD51を発行して任意の1データ・ブロックをメモリの所定の位置にロードするルーチンSDC_

READ_SINGLE_POLが収められています．割り込み機能は使用せず，ポーリングでオペレーション終了をチェックします．

(2) FATファイル・システムのアクセス

以上のルーチンを使用してSDメモリ・カード上のFATファイル・システムからデータを読み出すのがfat.asm, fat_acc.asmに収められたルーチン群です．

SC_FAT_INIT (fat_acc.asm内)は，SDメモリ・カード上のFATファイル・システムの仕様から，ファイルへアクセスするための情報を取得します．**図8**にFATファイル・システムの構造を簡単に示します[8]．FATファイル・システムでは，記憶領域をクラスタという1Kバイト～64Kバイトの大きさの単位で扱います．ファイルは一つまたは複数のクラスタで構成され，複数クラスタの場合は，あるクラスタの次のクラスタがどれか，という情報を記載したテーブル(FAT)を参照することによってファイルを構成するすべてのクラスタにアクセスすることができます．

FATファイル・システムからファイルを読み出すのに必要な情報は，

- FATの先頭アドレス(変数SD_FAT_FATに保存)
- ルート・ディレクトリ・エントリの先頭アドレス(変数SD_FAT_RDEA)
- ルート・ディレクトリ・エントリの大きさ(変数SD_FAT_RDES)
- データ領域の先頭アドレス(変数SD_FAT_DAB)
- クラスタの大きさ(変数DA_FAT_CLS)

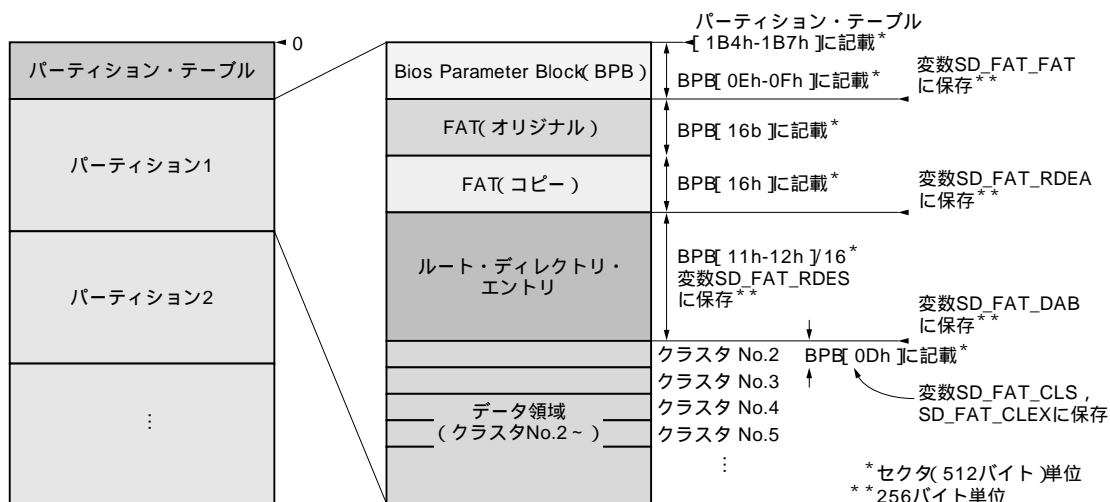
です．それぞれの計算方法については**図8**と参考文献[8]を参照してください．ここでは，今回作成したルーチン群が，これらの変数を使ってFAT16ファイル・システムからどのようにファイルを読み出すかについて説明します．

ファイル・システムに収録されているファイル一覧は，ルート・ディレクトリ・エントリに記されています．この領域には先頭からファイルの名前や属性を記録した大きさ20hバイトのレコードが並んでいます．各レコードの先頭から0Bhバイトがファイル名と拡張子，1Ahバイト目から2バイトがファイルの先頭のクラスタの番号です．変数SD_FAT_DAB ＋ 変数SD_FAT_CLS * (クラスタ番号－2)によりクラスタ番号からアドレスを知ることができます．この計算をするルーチンがSC_FAT_CL2LBA (fat_acc.asm内)です．

次のクラスタ番号を知るには，FATを参照します．FATには，先頭から(クラスタ番号×2)バイト目からの2バイトに，次のクラスタ番号が記されています．これを参照するルーチンがSC_FAT_CLCHAIN (fat_acc.asm内)です．このルーチンはメモリ・カードに収められているFATの必要な部分を1セクタ(＝512バイト)分，最初に述べたバッファSD_FAT_BUFFERに読み出して利用します．

これら二つのルーチンは，fat.asm内のFAT_READ_BLOCKから呼び出されます．このルーチンは，繰り返しコールすることで，指定されたクラスタからファイルの後尾に向かって順次，1セクタずつ読み出していくことができます．このルーチンでは，変数CUR_SECTとCUR_CLUSに，ファイル中の現在位置を記録しています．これらはSD_FAT_READ_BLOCK_INITというルーチンに，

図8 FATファイル・システムの構造

記憶領域をクラスタという1Kバイト～64Kバイトの大きさの単位で扱う．ファイルは一つまたは複数のクラスタで構成され，複数クラスタの場合は，あるクラスタの次のクラスタがどれか，という情報を記載したテーブル(FAT)を参照することによってファイルを構成するすべてのクラスタにアクセスすることができる．

先頭クラスタを与えることで初期化できます．

まとめると，まずSC_FAT_INITでファイル・システムの情報を取得，ルート・ディレクトリ・エントリから読み出したいファイルの先頭クラスタを探し出してSD_FAT_READ_BLOCK_INITに与えることでファイル読み出しの準備が完了，後はSD_FAT_READ_BLOCKを繰り返し呼び出すことで順次ファイルの内容をバッファに読み出していくことができます．

● 今後の拡張

今回はなるべく簡単な仕様としたので，拡張すべき点はたくさんあります．

まずあげられるのはオーバサンプリングです．これはデータのサンプル数を何倍かに増やし，空白だったサンプルに0値データを挿入し，ローパス・フィルタにより高調波成分を除去することで周波数成分がもとのデータとまったく変わらず，サンプリング周波数だけが高いデータを得る手法です．多くの場合，この処理にはFIRフィルタが用いられますが，その実装に必要な乗算器はSpartan-3Eにはもともと内蔵されています．さらに，Core GeneratorではFIRフィルタを自動生成してくれる機能もあるので，FPGAリソースに余裕があれば比較的簡単に実現できると思われます．例えば16倍のオーバサンプリングをかけると，350kHz程度までクリアですから，最初の方に述べたようにローパス・フィルタに*CR*のみの受動フィルタを用いることができ，音質向上に大きく寄与します．

また，D-Aコンバータの並列化によりノイズ・レベルを抑制する，*I-V*コンバータをOPアンプでなく，ディスクリート部品を用いたカレント・ミラー回路などで構成する，プリント基板を製作する，電源をディジタル，アナログそれぞれに用意する，D-AコンバータICに与えるクロックを高精度なものに変更するなど，さまざまな拡張により音質の改善を図ることができます．

ユーザ・インターフェースの拡張も考えられます．例えば，一時停止や早送りなどの機能，最終的には再生リストの管理などまで行えるとよいと思っています．

前述のように，本誌に収録されているデータは最初の1曲のみの再生にしか対応していません．ルート・ディレクトリにあるすべてのファイルを順次再生することのできるものを後ほど公開します．本誌Webページ（http://www.cqpub.co.jp/dwm/）か，筆者のWebページ（http://www.digicat.info/dcframe.cgi?dwm）をご覧ください．

最後に，各部品の入手性について紹介しておきます．最も入手性が悪かったのは16.9344MHzの水晶発振器です．唯一見つけられたのがサンエレクトロの通信販売（http://www1.odn.ne.jp/~aal22410/）でした．D-Aコンバータは比較的簡単に入手できます．PCM1702はRSコンポーネンツとDigi-Keyで，PCM1704はDigi-Keyと共立電子の店頭で，PCM56はRSコンポーネンツ，Digi-Key，若松電子の通販（Kグレードあり）で入手可能です．いずれも2,000～4,000円程度です．また，*I-V*コンバータで使用したOPアンプOPA627はRSコンポーネンツ，サンエレクトロなどで入手可能ですが，1個あたり3,000円以上とかなりいい値段がします．こだわらないのならばもっと安いものでも十分です．いろいろ調べたところ，AD797，AD823，AD825，AD827，NJM4580，NJM2114，OPA2604などが人気があるようです．

参考・引用*文献

(1) Texas Instruments；Super HiFi DDAC Boardの設計（EVM-1702の技術解説），Application Note SBAA061，http://www.tij.co.jp/jsc/psheets/SBAA061.pdf
(2) 河野 崇；簡易シリアル端末の制作，Design Wave Magazine 2003年10月号．
(3) 河野 崇；luna0 Developper's Manual，http://www.digicat.info/softwares/luna0.pdf
(4) SD Card Association；SD Specifications Part1 Physical Layer Simlified Specification，Version 2.00，http://www.sdcard.org/sd_memorycard/index.html，Sep．2006．
(5) SanDisk；SanDisk MultiMediaCard and Reduced-Size MultiMediaCard，Product Manual Version 1.3，http://www.sandisk.com/Oem/Manuals/，Apr．2005．
(6) インターフェース編集部編；フラッシュ・メモリ・カードの徹底研究，第2部 SD/MMCカード編，pp.70-154，CQ出版．
(7) 特集 実験研究！大容量メモリカード，トランジスタ技術，2007年2月号．
(8) Operating Systems，Filesystems，FATFS，http://perso.orange.fr/pierrelib/filesystems/fat16.html

こうの・たかし
東京大学生産技術研究所准教授

<筆者プロフィール>
河野 崇．医師，工学博士．1996年3月，東京大学医学部医学科卒．2002年3月，東京大学工学系研究科（計数工学専攻）にて工学博士号取得．2006年9月より現職へ．専門は電子回路ニューロンを中心としたニューロモルフィックハードウェアの設計．学部時代よりFPGAを用いた回路設計に趣味と実益をかねて携わる．

8